Package ‘exuber’

July 15, 2019

Type Package

Title Econometric Analysis of Explosive Time Series

Version 0.3.0

Description Testing for and dating periods of explosive
dynamics (exuberance) in time series using the univariate and panel
recursive unit root tests proposed by Phillips et al. (2015)
algorithm utilizes the matrix inversion lemma to avoid matrix
inversion which results in significant speed improvements. Simulation
of a variety of periodically-collapsing bubble processes.

License GPL-3

URL https://github.com/kvasilopoulos/exuber

BugReports https://github.com/kvasilopoulos/exuber/issues

Depends R (>= 3.0.2)

Imports doSNOW (>= 1.0.16), dplyr (>= 0.8.0.1), foreach (>= 1.4.4),
ggplot2 (>= 3.1.1), grid, gridExtra (>= 2.3), lubridate (>=
1.7.4), parallel, purrr (>= 0.3.2), Rcpp (>= 0.12.17), rlang
(>= 0.3.4), tibble (>= 2.1.1), cli (>= 1.1.0), generics (>=
0.0.2), tidyr (>= 0.8.3), glue (>= 1.3.1), zoo (>= 1.8.5)

Suggests covr (>= 3.2.1), knitr (>= 1.22), rmarkdown (>= 1.12),
spelling (>= 2.1), testthat (>= 2.1.1), withr (>= 2.1.2),
stringr (>= 1.4.0)

LinkingTo Rcpp (>= 1.0.1), RcppArmadillo (>= 0.9.400.2.0)

VignetteBuilder knitr

Encoding UTF-8

LazyData true

Language en-US

RoxygenNote 6.1.1

NeedsCompilation yes
**augment_join**

**Description**

Tidy a model of `radf` with a model of `cv`

**Usage**

```r
augment_join(x, y = NULL)
```
Arguments

x
An object of class radf

y
An object of class cv

Description
Plotting datestamp with geom_segment()

Usage

## S3 method for class 'datestamp'
applot(object, ...)

## S3 method for class 'datestamp'
fputy(model, data, ...)

Arguments

object
An object of class datestamp()

... further arguments passed to method. Specify common characteristics like ggplot2::xlab, that are later passed to ggplot chain. For multiple changes, the input in the argument should be in a list.

model
datestamp object

data
original dataset, not used (required by generic fortify() method).

Examples

dta <- cbind(sim_psy1(n = 100), sim_psy2(n = 100))

dta %>%
  radf() %>%
  datestamp() %>%
  autoplot()

# Change the colour manually
dta %>%
  radf() %>%
  datestamp() %>%
  autoplot() +
ggplot2::scale_colour_manual(values = rep("black", 4))
Description

Takes dist objects and returns a ggplot2 object

Usage

```r
## S3 method for class 'mc_distr'
autoplot(object, ...)

## S3 method for class 'wb_distr'
autoplot(object, ...)

## S3 method for class 'sb_distr'
autoplot(object, ...)
```

Arguments

- `object`: An `_dist` object.
- `...`: Additional arguments, used only in wb_distr facet options.

Description

`autoplot.radf` takes an radf object and returns a (list of) ggplot2 objects. `fortify.radf` takes an radf object and converts it into a data.frame. `ggarrange` is a wrapper of `arrangeGrob()`, which can be used directly after autoplot to place grobs on a page.

Usage

```r
## S3 method for class 'radf'
autoplot(object, cv = NULL, include = FALSE, 
          select = NULL, option = c("gsadf", "sadf"), min_duration = 0, 
          arrange = TRUE, ...)

## S3 method for class 'radf'
fortify(model, data, cv = NULL, include = FALSE, 
        select = NULL, option = c("gsadf", "sadf"), ...)

ggarrange(...)```
Arguments

object  An object of class \texttt{radf}.

cv      An object of class "cv". The output of \texttt{mc_cv}, \texttt{wb_cv} or \texttt{sb_cv}.

include If not FALSE, plot all variables regardless of rejecting the NULL at the 5% significance level.

select  If not NULL, only plot with names or column number matching this regular expression will be executed.

option  Whether to apply the "gsadf" or "sadf" methodology. Default is "gsadf".

min_duration The minimum duration of an explosive period for it to be reported. Default is 0.

arrange If FALSE returns a list of ggplot2 object, otherwise it grobs the plots on a single page.

... further arguments passed to method. Specify common characteristics like \texttt{ggplot2::xlab}, that are later passed to ggplot chain. For multiple changes, the input in the argument should be in a list.

model  An object of class \texttt{radf}.

data   original dataset, not used (required by generic \texttt{fortify} method).

Details

arrange offers flexibility to the user by specifying the desired output. If arrange = FALSE, the individual plots can be modified after creation the then rearranged with the \texttt{ggarrange} function into a single plot.

Examples

dta <- data.frame(psy1 = sim_psy1(n = 100), psy2 = sim_psy2(n = 100))

dta %>%
  radf() %>%
  autoplot(ncol = 2)

# For custom plotting with ggplot2
dta %>%
  radf() %>%
  fortify()
### calc_pvalue

**Calculate p-values**

**Description**

Calculate p-values

**Usage**

```r
calc_pvalue(x, dist = NULL)
```

**Arguments**

- `x`: An 'radf' object
- `dist`: Which type of distribution to use to calculate the p-values

**Examples**

```r
# Not run:
radf_psy1 <- radf(sim_psy1(100))
calc_pvalue(radf_psy1)

# Using the Wild-Bootstrapped
wb_psy1 <- wb_dist(sim_psy1(100))
calc_pvalue(radf_psy1, wb_psy1)
```

### col_names

**Retrieve/Set column names**

**Description**

Retrieve or set the column names of a class `radf()` object. Similar to `colnames`, with the only difference that `col_names` is for `radf()` objects.

**Usage**

```r
col_names(x, ...)

col_names(x) <- value
```
Arguments

- **x**: An object of class `radf()`
- **...**: Further arguments passed to methods.
- **value**: An ordered vector of the same length as the ‘index’ attribute of `x`.

Examples

```r
## Not run:
# Simulate bubble processes
da <- data.frame(psy1 = sim_psy1(n = 100), psy2 = sim_psy2(n = 100))

rfd <- radf(dta)
col_names(rfd) <- c("OneBubble", "TwoBubbles")

## End(Not run)
```

Description

A dataset containing simulated critical values for up to 600 observations based on default minimum window. The critical values have been simulated and stored as data to save computation time for the user. The stored critical values can be obtained with the `mc_cv()` function, using the `seed = 123`.

Usage

- `crit`

Format

A list with lower level lists that contain

- **adf_cv**: Augmented Dickey-Fuller
- **badf_cv**: Backward Augmented Dickey-Fuller
- **sadf_cv**: Supremum Augmented Dickey-Fuller
- **bsadf_cv**: Backward Supremum Augmented Dickey-Fuller
- **gsadf_cv**: Generalized Supremum Augmented Dickey Fuller

Source

simulated from exuber package function `mc_cv()`

Examples

```r
## Not run:
all.equal(crit[[50]], mc_cv(50, seed = 123))

## End(Not run)
```
**datestamp**

*Date-stamping periods of mildly explosive behavior*

**Description**

Computes the origination, termination and duration of episodes during which the time series display explosive dynamics.

**Usage**

```r
datestamp(object, cv = NULL, option = c("gsadf", "sadf"),
          min_duration = 0)
```

**Arguments**

- **object**: An object of class `radf()`.
- **cv**: An object of class "cv". The output of `mc_cv()`, `wb_cv()` or `sb_cv()`
- **option**: Whether to apply the "gsadf" or "sadf" methodology. Default is "gsadf".
- **min_duration**: The minimum duration of an explosive period for it to be reported. Default is 0.

**Details**

Datestamp also stores a vector in 0,1 that corresponds to reject, accept respectively, for all series in the time period. This output can be used as a dummy that indicates the occurrence of a bubble.

Setting `min_duration` removes very short episode of exuberance. Phillips et al. (2015) propose two simple rules of thumb to remove short periods of explosive dynamics, "log(T)/T", where T is the number of observations.

**Value**

Returns a list of values for each explosive sub-period, giving the origin and termination dates as well as the number of periods explosive behavior lasts.

**References**

**Diagnostics**

**Description**

Finds the series that reject the null for at the 5% significance level.

**Usage**

```r
diagnostics(object, cv = NULL, option = c("gsadf", "sadf"))
```

**Arguments**

- **object** An object of class `radf()`.
- **cv** An object of class "cv". The output of `mc_cv()`, `wb_cv()` or `sb_cv()`
- **option** Whether to apply the "gsadf" or "sadf" methodology. Default is "gsadf".

**Details**

Diagnostics also stores a vector in 0,1 that corresponds to reject, accept respectively.

**Value**

Returns a list with the series that reject and the series that do not reject the Null Hypothesis.

---

**index.radf**

**Retrieve/Replace the index**

**Description**

Retrieve or replace the index of an `radf` object.

**Usage**

```r
## S3 method for class 'radf'
index(x, trunc = FALSE, ...)

## S3 replacement method for class 'radf'
index(x) <- value
```

**Arguments**

- **x** An object of class `radf()`
- **trunc** default FALSE. If TRUE the index formed by truncating the value in the minimum window.
- **...** Further arguments passed to methods.
- **value** An ordered vector of the same length as the ‘index’ attribute of x.
Details

If the user does not specify an index for the estimation a pseudo-index is generated which is a sequential numeric series. After the estimation, the user can use index to retrieve or `index<-` to replace the index. The index can be either numeric or Date.

---

**mc_cv**

*Monte Carlo Critical Values*

---

**Description**

`mc_cv` computes Monte Carlo critical values for the recursive unit root tests. `mc_dist` computes the distribution.

**Usage**

```r
mc_cv(n, minw = NULL, nrep = 2000, seed = NULL, opt_badf = c("fixed", "asymptotic", "simulated"), opt_bsadf = c("conservative", "conventional"))

mc_distr(n, minw = NULL, nrep = 2000, seed = NULL)
```

**Arguments**

- `n` A positive integer. The sample size.
- `minw` A positive integer. The minimum window size, which defaults to \((0.01 + 1.8/\sqrt{T})) \times T\).
- `nrep` A positive integer. The number of Monte Carlo simulations.
- `seed` An object specifying if and how the random number generator(rng) should be initialized. Either NULL or an integer will be used in a call to `setNseed` before simulation. If set, the value is saved as the "seed" attribute of the returned value. The default, NULL will not change the rng state, and return .Random.seed as the "seed" attribute.
- `opt_badf` Options for badf critical value calculation. "fixed" corresponds to \(\log(\log(n*s))/100\) rule, "asymptotic" to asymptotic critical values and simulated to the monte carlo simulations.
- `opt_bsadf` Options for bsadf critical value calculation. "conventional" corresponds to the max of the quantile of the simulated distribution, while "conservative" corresponds to the quantile of the max which is more conservative in nature, thus the name.

**Value**

A list that contains the critical values for ADF, BADF, BSADF and GSADF t-statistics.
psy_minw

See Also

wb_cv for Wild Bootstrapped critical values and sb_cv for Sieve Bootstrapped critical values

Examples

```r
## Not run:
# Default minimum window
mc <- mc_cv(n = 100)

# Change the minimum window and the number of simulations
mc <- mc_cv(n = 100, nrep = 2500, minw = 20)

mdist <- mc_distr(n = 100)
autoplot(mdistr)

## End(Not run)
```

psy_minw  

*Helper functions in accordance to PSY(2015)*

Description

psy_minw proposes a minimum window and psy_ds proposes a rule of thumb to exclude periods of exuberance.

Usage

```r
psy_minw(n)

psy_ds(n, rule = 1, delta = 1)
```

Arguments

- `n` A positive integer. The sample size.
- `rule` Rule 1 corresponds to \( \log(T) \), while rule 2 \( \log(T)/T \)
- `delta` Frequency-dependent parameter

Details

delta depends on the frequency of the data and the minimal duration condition. For example, for a 30-year period, we set arbitrarily duration to exceed periods such as one year. Then, delta should be 0.7 for yearly data and 5 for monthly data.

References

Examples

```r
psy_minw(100)
psy_ds(100)
```

---

**radf**  
*Recursive Augmented Dickey-Fuller Test*

**Description**

`radf` returns the t-statistics from a recursive Augmented Dickey-Fuller test.

**Usage**

```r
radf(data, minw = NULL, lag = 0)
```

**Arguments**

- **data**  
  A univariate or multivariate numeric `ts` object, `data.frame` or `matrix`. The estimation process cannot handle NA values.

- **minw**  
  A positive integer. The minimum window size, which defaults to \((0.01 + 1.8/\sqrt{T}) \times T\).

- **lag**  
  A non-negative integer. The lag of the Augmented Dickey-Fuller regression.

**Value**

A list that contains the t-statistic (sequence) for:

- **adf**  
  Augmented Dickey-Fuller

- **badf**  
  Backward Augmented Dickey-Fuller

- **sadf**  
  Supremum Augmented Dickey-Fuller

- **bsadf**  
  Backward Supremum Augmented Dickey-Fuller

- **gsadf**  
  Generalized Supremum Augmented Dickey-Fuller

**References**


Examples

```r
# Simulate bubble processes
dta <- data.frame(psy1 = sim_psy1(n = 100), psy2 = sim_psy2(n = 100))

rfd <- radf(dta)

# For lag = 1 and minimum window = 20
rfd <- radf(dta, minw = 20, lag = 1)
```

---

**sb_cv**

*Panel Sieve Bootstrap Critical Values*

**Description**

`sb_cv` computes p-values for the panel recursive unit root test using the sieve bootstrap procedure outlined in Pavlidis et al. (2016). `sb_dist` computes the distribution.

**Usage**

```r
sb_cv(data, minw = NULL, lag = 0, nboot = 1000, seed = NULL)
sb_distr(data, minw = NULL, lag = 0, nboot = 1000, seed = NULL)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>A univariate or multivariate numeric ts object, data.frame or matrix. The estimation process cannot handle NA values.</td>
</tr>
<tr>
<td>minw</td>
<td>A positive integer. The minimum window size, which defaults to ((0.01 + 1.8 / \sqrt{T}) \times T).</td>
</tr>
<tr>
<td>lag</td>
<td>A non-negative integer. The lag of the Augmented Dickey-Fuller regression.</td>
</tr>
<tr>
<td>nboot</td>
<td>A positive integer indicating the number of bootstraps. Default is 1000 repetitions.</td>
</tr>
<tr>
<td>seed</td>
<td>An object specifying if and how the random number generator(rng) should be initialized. Either NULL or an integer will be used in a call to set.seed before simulation. If set, the value is save as &quot;seed&quot; attribute of the returned value. The default, NULL will not change the rng state, and return .Random.seed as the &quot;seed&quot; attribute.</td>
</tr>
</tbody>
</table>

**Value**

A list that contains the panel critical values for BSADF and GSADF t-statistics.
References


See Also

mc_cv for Monte Carlo critical values and wb_cv for Wild Bootstrapped critical values

Examples

## Not run:

# Simulate bubble processes
set.seed(4441)
dta <- data.frame(  
  "psy1" = sim_psy1(100),  
  "psy2" = sim_psy2(100),  
  "evans" = sim_evans(100),  
  "div" = sim_div(100),  
  "blan" = sim_blan(100)
)

# Panel critical vales should have the same lag length with the estimation
sb <- sb_cv(dta, lag = 1)
dta %>%
  radf(lag = 1) %>%
  summary(cv = sb)
dta %>%
  radf(lag = 1) %>%
  autoplot(cv = sb)

# Simulate distribution
sb_dist(dta, lag = 1)

## End(Not run)

---

**sim_blan**  
*Simulation of a Blanchard (1979) bubble process*

**Description**

Simulation of a Blanchard (1979) rational bubble process.

**Usage**

```r
sim_blan(n, pi = 0.7, sigma = 0.03, r = 0.05, b0 = 0.1,  
seed = NULL)
```
**Arguments**

- **n**
  A strictly positive integer specifying the length of the simulated output series.

- **pi**
  A positive value in (0, 1) which governs the probability of the bubble continuing to grow.

- **sigma**
  A positive scalar indicating the standard deviation of the innovations.

- **r**
  A positive scalar that determines the growth rate of the bubble process.

- **b0**
  The initial value of the bubble

- **seed**
  An object specifying if and how the random number generator(rng) should be initialized. Either NULL or an integer will be used in a call to set.seed before simulation. If set, the value is saved as "seed" attribute of the returned value. The default, NULL will not change the rng state, and return .Random.seed as the "seed" attribute.

**Details**

Blanchard’s bubble process has two regimes, which occur with probability \( \pi \) and \( 1 - \pi \). In the first regime, the bubble grows exponentially, whereas in the second regime, the bubble collapses to a white noise.

With probability \( \pi \):

\[
B_{t+1} = \frac{1 + r}{\pi} B_t + \epsilon_{t+1}
\]

With probability \( 1 - \pi \):

\[
B_{t+1} = \epsilon_{t+1}
\]

where \( r \) is a positive constant and \( \epsilon \sim iid(0, \sigma^2) \).

**Value**

A numeric vector of length \( n \).

**References**


**See Also**

- `sim_psy1`, `sim_psy2`, `sim_evans`

**Examples**

```r
sim_blan(n = 100)
```
Description

Simulate (log) dividends from a random walk with drift.

Usage

\[
\text{sim\_div}(n, \mu, \sigma, r = 0.05, \log = \text{FALSE}, \text{output} = \text{c("pf", "d")}, \\
\text{seed} = \text{NULL})
\]

Arguments

- \(n\) A strictly positive integer specifying the length of the simulated output series.
- \(\mu\) A scalar indicating the drift.
- \(\sigma\) A positive scalar indicating the standard deviation of the innovations.
- \(r\) A positive value indicating the discount factor.
- \(\log\) A logical. If true dividends follow a lognormal distribution.
- \(\text{output}\) A character string giving the fundamental price("pf") or dividend series("d"). Default is 'pf'.
- \(\text{seed}\) An object specifying if and how the random number generator(rng) should be initialized. Either NULL or an integer will be used in a call to set.seed before simulation. If set, the value is save as "seed" attribute of the returned value. The default, NULL will note change the rng state, and return .Random.seed as the "seed" attribute.

Details

If \(\log\) is set to FALSE (default value) the dividends follow:

\[
d_t = \mu + d_{t-1} + \epsilon_t
\]

where \(\epsilon \sim \mathcal{N}(0, \sigma^2)\). The default parameters are \(\mu = 0.0373, \sigma^2 = 0.1574\) and \(d[0] = 1.3\) (the initial value of the dividend sequence). The above equation can be solved to yield the fundamental price:

\[
F_t = \mu(1 + r)^{-2} + r^{-1}d_t
\]

If \(\log\) is set to TRUE then dividends follow a lognormal distribution or \log(dividends) follow:

\[
\ln(d_t) = \mu + \ln(d_{t-1}) + \epsilon_t
\]

where \(\epsilon \sim \mathcal{N}(0, \sigma^2)\). Default parameters are \(\mu = 0.013, \sigma^2 = 0.16\). The fundamental price for this case is:
\[
F_t = \frac{1 + g}{r - g} d_t
\]

where \(1 + g = \exp(\mu + \sigma^2/2)\). All default parameter values are those suggested by West (1988).

Value

A numeric vector of length \(n\).

References


Examples

\# Price is the sum of the bubble and fundamental components
\# 20 is the scaling factor
pf <- sim_div(100, r = 0.05, output = "pf")
pb <- sim_evans(100, r = 0.05)
p <- pf + 20 * pb

---

**sim_evans**  
*Simulation of an Evans (1991) bubble process*

**Description**

Simulation of an Evans (1991) rational periodically collapsing bubble process.

**Usage**

\[
sim_evans(n, alpha = 1, delta = 0.5, tau = 0.05, pi = 0.7, r = 0.05, b1 = delta, seed = NULL)
\]

**Arguments**

- \(n\): A strictly positive integer specifying the length of the simulated output series.
- \(alpha\): A positive scalar, with restrictions (see details).
- \(delta\): A positive scalar, with restrictions (see details).
- \(tau\): The standard deviation of the innovations.
- \(pi\): A positive value in (0, 1) which governs the probability of the bubble continuing to grow.
- \(r\): A positive scalar that determines the growth rate of the bubble process.
- \(b1\): A positive scalar, the initial value of the series. Defaults to \(delta\).
seed  An object specifying if and how the random number generator (rng) should be initialized. Either NULL or an integer will be used in a call to set.seed before simulation. If set, the value is saved as "seed" attribute of the returned value. The default, NULL will not change the rng state, and return .Random.seed as the "seed" attribute.

Details
delta and alpha are positive parameters which satisfy $0 < \delta < (1 + r)\alpha$. delta represents the size of the bubble after collapse. The default value of $r$ is 0.05. The function checks whether alpha and delta satisfy this condition and will return an error if not.

The Evans bubble has two regimes. If $B_t \leq \alpha$ the bubble grows at an average rate of $1 + r$:

$$B_{t+1} = (1 + r)B_t u_{t+1},$$

When $B_t > \alpha$ the bubble expands at an increased rate of $(1 + r)\pi^{-1}$:

$$B_{t+1} = \left[\delta + (1 + r)\pi^{-1}\theta_{t+1}(B_t - (1 + r)^{-1}\delta B_t)\right]u_{t+1},$$

where $\theta$ is an indicator function taking a value of 0 with probability $1 - \pi$ and 1 with probability $\pi$. In this secondary phase there is a probability $(1 - \pi)$ that the bubble collapses to delta and the process starts again. By modifying the values of delta, alpha and pi the user can change the frequency at which bubbles appear, the mean duration of a bubble before collapse and the scale of the bubble.

Value
A numeric vector of length n.

References

See Also
sim_psy1, sim_psy2, sim_blan

---

**sim_psy1**  
*Simulation of a single-bubble process*

Description
The following function generates a time series which switches from a martingale to a mildly explosive process and then back to a martingale.
Usage

\( \text{sim}_{\text{psy}1}(n, \text{te} = 0.4 \times n, \text{tf} = 0.15 \times n + \text{te}, c = 1, \text{alpha} = 0.6, \text{sigma} = 6.79, \text{seed} = \text{NULL}) \)

Arguments

- \( n \): A strictly positive integer specifying the length of the simulated output series.
- \( \text{te} \): A scalar in \((0, \text{tf})\) specifying the observation in which the bubble originates.
- \( \text{tf} \): A scalar in \((\text{te}, n)\) specifying the observation in which the bubble collapses.
- \( c \): A positive scalar determining the autoregressive coefficient in the explosive regime.
- \( \text{alpha} \): A positive scalar in \((0, 1)\) determining the value of the expansion rate in the autoregressive coefficient.
- \( \text{sigma} \): A positive scalar indicating the standard deviation of the innovations.
- \( \text{seed} \): An object specifying if and how the random number generator(rng) should be initialized. Either NULL or an integer will be used in a call to set. seed before simulation. If set, the value is save as "seed" attribute of the returned value. The default, NULL will note change the rng state, and return .Random.seed as the "seed" attribute.

Details

The data generating process is described by the following equation:

\[
X_t = X_{t-1} \mathbb{1}_{t < \tau_e} + \delta_T X_{t-1} \mathbb{1}_{\tau_e \leq t \leq \tau_f} + \left( \sum_{k=\tau_f+1}^{t} \epsilon_k + X^*_{\tau_f} \right) \mathbb{1}_{t > \tau_f} + \epsilon_t \mathbb{1}_{t \leq \tau_f}
\]

where the autoregressive coefficient \( \delta_T \) is given by:

\[
\delta_T = 1 + c T^{-\alpha}
\]

with \( c > 0, \alpha \in (0, 1), \epsilon \sim iid(0, \sigma^2) \) and \( X_{\tau_f} = X_{\tau_e} + X^* \). During the pre- and post- bubble periods, \( N_0 = [1, \tau_e) \), \( X \) is a pure random walk process. During the bubble expansion period \( B = [\tau_e, \tau_f] \) is a mildly explosive process with expansion rate given by the autoregressive coefficient \( \delta_T \), and continues its martingale path for the subsequent period \( N_1 = (\tau_f, \tau] \).

For further details the user can refer to Phillips et al. (2015) p. 1054.

Value

A numeric vector of length \( n \).

References

See Also

`sim_psy2`, `sim_blan`, `sim_evans`

Examples

```r
# 100 periods with bubble origination date 40 and termination date 55
sim_psy1(n = 100)

# 200 periods with bubble origination date 80 and termination date 110
sim_psy1(n = 200)

# 200 periods with bubble origination date 100 and termination date 150
sim_psy1(n = 200, te = 100, tf = 150)
```

---

`sim_psy2`  
**Simulation of a two-bubble process**

Description

The following data generating process is similar to `sim_psy1`, with the difference that there are two episodes of mildly explosive dynamics.

Usage

```r
sim_psy2(n, te1 = 0.2 * n, tf1 = 0.2 * n + te1, te2 = 0.6 * n,
         tf2 = 0.1 * n + te2, c = 1, alpha = 0.6, sigma = 6.79,
         seed = NULL)
```

Arguments

- `n`  
  A strictly positive integer specifying the length of the simulated output series.

- `te1`  
  A scalar in (0, n) specifying the observation in which the first bubble originates.

- `tf1`  
  A scalar in (te1, n) specifying the observation in which the first bubble collapses.

- `te2`  
  A scalar in (tf1, n) specifying the observation in which the second bubble originates.

- `tf2`  
  A scalar in (te2, n) specifying the observation in which the second bubble collapses.

- `c`  
  A positive scalar determining the autoregressive coefficient in the explosive regime.

- `alpha`  
  A positive scalar in (0, 1) determining the value of the expansion rate in the autoregressive coefficient.

- `sigma`  
  A positive scalar indicating the standard deviation of the innovations.

- `seed`  
  An object specifying if and how the random number generator (rng) should be initialized. Either `NULL` or an integer will be used in a call to `set.seed` before simulation. If set, the value is save as "seed" attribute of the returned value. The default, `NULL` will not change the rng state, and return `.Random.seed` as the "seed" attribute.
The data generating process is described by:

\[ X_t = X_{t-1}1\{t \in N_0\} + \delta_T X_{t-1}1\{t \in B_1 \cup B_2\} + \left( \sum_{k=\tau_f+1}^{t} \epsilon_k + X^*_{\tau_f} \right) 1\{t \in N_1\} \]

\[ + \left( \sum_{l=\tau_f+1}^{t} \epsilon_l + X^*_{\tau_f} \right) 1\{t \in N_2\} + \epsilon_t 1\{t \in N_0 \cup B_1 \cup B_2\} \]

where the autoregressive coefficient \( \delta_T \) is given:

\[ \delta_T = 1 + cT^{-\alpha} \]

with \( c > 0, \alpha \in (0, 1), \epsilon \sim iid(0, \sigma^2) \), \( X_{\tau_1} = X_{\tau_1 e} + X^* \) and \( X_{\tau_2} = X_{\tau_2 e} + X^* \). We use the notation \( N_0 = [1, \tau_1 e] \), \( B_1 = [\tau_1 e, \tau_1 f] \), \( N_1 = (\tau_1 f, \tau_2 e) \), \( B_2 = [\tau_2 e, \tau_2 f] \), \( N_2 = (\tau_2 f, \tau) \), where \( \tau \) is the last observation of the sample. After the collapse of the first bubble, \( X_t \) resumes a martingale path until time \( \tau_{2e} - 1 \), and a second episode of exuberance begins at \( \tau_{2e} \). The expansion process lasts until \( \tau_2 f \) and collapses to a value of \( X^*_{\tau_2 f} \). The process then continues on a martingale path until the end of the sample period \( \tau \). The expansion duration of the first bubble is assumed to be longer than that of the second bubble, i.e. \( \tau_1 f - \tau_{1e} > \tau_2 f - \tau_{2e} \).

For further details the user can refer to Phillips et al., (2015) p. 1055.

Value

A numeric vector of length \( n \).

References


See Also

\( \text{sim_psy1, sim_blan, sim_evans} \)

Examples

# 100 periods with bubble origination dates 20/60 and termination dates 40/70 respectively sim_psy2(n = 100)

# 200 periods with bubble origination dates 40/120 and termination dates 80/140 respectively sim_psy2(n = 200)
Summary

Summarizing radf models

Description

summary method for class "radf"

Usage

```r
## S3 method for class 'radf'
summary(object, cv = NULL, ...)
```

Arguments

- `object`: An object of class `radf()`.
- `cv`: An object of class "cv". The output of `mc_cv()`, `wb_cv()` or `sb_cv()`
- `...`: further arguments passed to methods, not used.

Value

Returns a list of summary statistics, the t-statistic and the critical values of the ADF, SADF and GSADF.

Examples

```r
# Simulate bubble processes, compute the t-stat and critical values
set.seed(4441)
dta <- data.frame(psy1 = sim_psy1(n = 100), psy2 = sim_psy2(n = 100))
rfd <- radf(dta)

# Summary, diagnostics and datestamp (default)
summary(rfd)
diagnostics(rfd)
datestamp(rfd)

#' # Diagnostics for 'sadf'
diagnostics(rfd, option = "sadf")

# Use log(T)/T rule of thumb to omit periods of explosiveness which are short-lived
rot <- round(log(nrow(rfd)) / NROW(rfd))
datestamp(rfd, min_duration = rot)
## Not run:
# Summary, diagnostics and datestamp (Wild Bootstrapped critical values)

wb <- wb_cv(dta)

summary(rfd, cv = wb)
diagnostics(rfd, cv = wb)
```
tidy.mc_cv

```r
datestamp(rfd, cv = wb)
## End(Not run)
```

---

**tidy.mc_cv**  
*Tidy an cv object*

**Description**

Tidy an cv object

**Usage**

```r
## S3 method for class 'mc_cv'
tidy(x, format = c("wide", "long"), ...)

## S3 method for class 'wb_cv'
tidy(x, format = c("wide", "long"), ...)

## S3 method for class 'sb_cv'
tidy(x, format = c("wide", "long"), ...)

## S3 method for class 'mc_cv'
augment(x, format = c("wide", "long"), ...)

## S3 method for class 'wb_cv'
augment(x, format = c("wide", "long"), ...)

## S3 method for class 'sb_cv'
augment(x, format = c("wide", "long"), ...)
```

**Arguments**

- `x`  
  An cv object

- `format`  
  Long or wide format

- `...`  
  Additional arguments. Not used.

**Value**

A `tibble::tibble()`

- `sig` The significance level.
- `name` The name of the series (when format is "long")
- `crit` The critical value (when format is "long")
Examples

```r
## Not run:
mc <- mc_cv(100)

# Get the critical values
tidy(mc)

# Get the critical value sequences
augment(mc)

## End(Not run)
```

### tidy.mc_distr

Tidying *_dist objects

Description

tidy *_dist takes an mc_distr, wb_distr or sb_distr object and returns a tibble.

Usage

```r
## S3 method for class 'mc_distr'
tidy(x, ...)

## S3 method for class 'wb_distr'
tidy(x, ...)

## S3 method for class 'sb_distr'
tidy(x, ...)
```

Arguments

- `x`: An *_dist object
- `...`: Additional arguments. Not used.

Value

A tibble::tibble()

Examples

```r
## Not run:
mc <- mc_cv(n = 100)
tidy(mc)

## End(Not run)
```
**tidy.radf**  
*Tidy an radf object*

**Description**
Tidy an radf object

**Usage**

```r
## S3 method for class 'radf'
tidy(x, format = c("wide", "long"), ...)
```

```r
## S3 method for class 'radf'
augment(x, format = c("wide", "long"), panel = FALSE, ...)
```

```r
## S3 method for class 'radf'
glance(x, format = c("wide", "long"), ...)
```

**Arguments**

- `x` An radf object
- `format` Long or wide format
- `...` Additional arguments. Not used.
- `panel` Either univariate or panel bsadf.

**Value**

A `tibble::tibble()`

**Examples**

```r
## Not run:
dta <- data.frame(psy1 = sim_psy1(n = 100), psy2 = sim_psy2(n = 100))

rfd <- radf(dta)

# Get the t-stat
tidy(rfd)

# Get the t-stat sequences
augment(rfd)

# Get the panel t-stat
glance(rfd)

## End(Not run)
```
**wb_cv**

**Wild Bootstrap Critical values**

**Description**

wb_cv performs the Harvey et al. (2016) wild bootstrap re-sampling scheme, which is asymptotically robust to non-stationary volatility, to generate critical values for the recursive unit root tests. wb_dist computes the distribution.

**Usage**

```r
wb_cv(data, minw = NULL, nboot = 1000, dist_rad = FALSE, seed = NULL)
wb_distr(data, minw = NULL, nboot = 1000, dist_rad = FALSE, seed = NULL)
```

**Arguments**

- `data` A univariate or multivariate numeric ts object, data.frame or matrix. The estimation process cannot handle NA values.
- `minw` A positive integer. The minimum window size, which defaults to \((0.01 + 1.8/\sqrt{T)}) \times T\).
- `nboot` A positive integer indicating the number of bootstraps. Default is 1000 repetitions.
- `dist_rad` Logical. If TRUE then the Rademacher distribution will be used.
- `seed` An object specifying if and how the random number generator(rng) should be initialized. Either NULL or an integer will be used in a call to set.seed before simulation. If set, the value is save as "seed" attribute of the returned value. The default, NULL will not change the rng state, and return .Random.seed as the "seed" attribute.

**Details**

This approach involves applying a wild bootstrap re-sampling scheme to construct the bootstrap analogue of the Phillips et al. (2015) test which is asymptotically robust to non-stationary volatility.

**Value**

A list that contains the critical values for ADF, BADF, BSADF and GSADF t-statistics.
References


See Also

- `mc_cv` for Monte Carlo critical values and `sb_cv` for Sieve Bootstrapped critical values

Examples

```r
## Not run:
# Simulate bubble processes
dta <- data.frame(psy1 = sim_psy1(n = 100), psy2 = sim_psy2(n = 100))

# Default minimum window
wb <- wb_cv(dta)

# Change the minimum window and the number of bootstraps
wb <- wb_cv(dta, nboot = 1500, minw = 20)

# Simulate distribution
wb_distr(dta)

## End(Not run)
```
Index

*Topic datasets
crit, 7

arrangeGrob(), 4
augment.mc_cv (tidy.mc_cv), 23
augment.radf (tidy.radf), 25
augment.sb_cv (tidy.mc_cv), 23
augment.wb_cv (tidy.mc_cv), 23
augment_join, 2
autoplot.datestamp, 3
autoplot.mc_distr, 4
autoplot.radf, 4
autoplot.sb_distr (autoplot.mc_distr), 4
autoplot.wb_distr (autoplot.mc_distr), 4
calc_pvalue, 6
col_names, 6
col_names<- (col_names), 6
crit, 7
datestamp, 8
datestamp(), 7
diagnostics, 9
fortify(), 3, 5
fortify.datestamp (autoplot.datestamp), 3
fortify.radf (autoplot.radf), 4
geom_segment(), 3
ggarrange (autoplot.radf), 4
glance.radf (tidy.radf), 25
index.radf, 9
index<- .radf (index.radf), 9
mc_cv, 10, 14, 27
mc_cv(), 5, 7–9, 22
mc_distr (mc_cv), 10
psy_ds (psy_minw), 11
psy_minw, 11
radf, 12
radf(), 5–9, 22
sb_cv, 11, 13, 27
sb_cv(), 5, 8, 9, 22
sb_distr (sb_cv), 13
sim_blan, 14, 18, 20, 21
sim_div, 16
sim_evans, 15, 17, 20, 21
sim_psy1, 15, 18, 18, 20, 21
sim_psy2, 15, 18, 20, 20
summary, 22
tibble::tibble(), 23–25
tidy.mc_cv, 23
tidy.mc_distr, 24
tidy.radf, 25
tidy.sb_cv (tidy.mc_cv), 23
tidy.sb_distr (tidy.mc_distr), 24
tidy.wb_cv (tidy.mc_cv), 23
tidy.wb_distr (tidy.mc_distr), 24
wb_cv, 11, 14, 26
wb_cv(), 5, 8, 9, 22
wb_distr (wb_cv), 26